Remarks

Applicants respectfully request reconsideration of the present application in view of the foregoing amendments and the following remarks. Claims 41-87 are pending in the application. No claims have been allowed. Claims 41-87 are rejected. These rejections are respectfully traversed. Claims 41, 51, 58, 63, 69, 72, 75, 79, and 84 are independent.

Patentability of Claims 41-87 over Schaumont under 35 U.S.C. 102

The Action rejects claims 41-87 under 35 U.S.C. 102(e) as being unpatentable over U.S. Patent 6,606,588 to Schaumont et al. ("Schaumont"). These rejections are respectfully traversed. For a 102(e) rejection to be proper, the applied art must show each and every element as set forth in a claim. (*See* MPEP § 2131.01) However, Schaumont fails to do so.

Independent Claim 41

Independent claim 41 recites:

> based upon a set of transformation rules, transforming plural methods of the interface into plural ports of a port map of a lower-level specification for a design unit.

Independent claim 41 is directed to transforming a programming language specification into a lower-level specification. The programming language specification includes an interface. Based upon a set of transformation rules, plural methods of the interface are transformed into plural ports of a port map of a lower-level specification for a design unit.

The Action relies on Schaumont disclosing the above-quoted language of claim 41. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 41. For example, at col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not involve "transforming plural methods of the interface" or "plural ports of a port map," and it does not teach or suggest "transforming plural methods of the interface into plural ports of a port map of a lower-level specification for a design unit," as recited in independent claim 41.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members (which are "pointers to communication queues" (col. 15, lines 21-22)), and "can optionally also be used to perform initialization of other private data." The parameters of the constructor include "a list of the queues that are used for communication." (Col. 15, line 29.) Schaumont goes on to describe "a firing rule and an operative part." Merely specifying communication queues for a block with private members (as in Schaumont) does not involve "transforming plural methods of the interface" or "plural ports of a port map," and it does not teach or suggest "transforming plural methods of the interface into plural ports of a port map of a lower-level specification for a design unit," as recited in independent claim 41. Similarly, including a list of communication queues as parameters for a constructor (as in Schaumont) also fails to teach or suggest the above-quoted language of claim 41.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation" to "obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. In this section, Schaumont references files 3.3 and 3.4, in which private members are declared for flow buffers, and a constructor includes parameters "_symb1" and "_symb2" for the flow buffers. Again, merely including a list of communication queues as parameters for a constructor (as in Schaumont) does not involve "transforming plural methods of the interface" or "plural ports of a port map," and it does not teach or suggest "transforming plural methods of the interface into plural ports of a port map of a lower-level specification for a design unit," as recited in independent claim 41.

Elsewhere, Schaumont describes using signal data types and a signal flowgraph "that collects all behavior that must be executed during one clock cycle," where the behavior contains "[a] set of inputs and outputs that relate signals to output and input queues." (Col. 25, lines 54-61.) "[A] signal flowgraph object connects local behavior (the signals) to the system through

communication queues. In hardware, the indication of input and output signals also results in ports on your resulting circuit." (Col. 25, lines 62-65; *see also* col. 38, lines 38-54.) Even if, for the sake of argument, Schaumont describes determining ports of a circuit, such determination uses signal data types and a detailed signal flowgraph (*id.*), which leads away from the above-quoted language of claim 41.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 41. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 41.

Dependent Claims 42-50

Claims 42-50 ultimately depend on independent claim 41. Thus, for at least the reasons set forth above with respect to independent claim 41, claims 42-50 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 42-50.

Independent Claim 51

Independent claim 51 recites:

> accepting a programming language specification, the programming language specification including plural calls to plural instances of a unit class, wherein a first call of the plural calls maps to a first instance of the plural instances of the unit class, wherein a second call of the plural calls maps to a second instance of the plural instances of the unit class, and wherein the programming language specification lacks explicit concurrency modeling for the plural instances of the unit class; and
>
> transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent execution of units represented by the plural instances of the unit class in the programming language specification.

Independent claim 51 is directed to transforming a programming language specification into a lower-level specification. The transforming includes generating lower-level description for handling concurrent execution of units represented by plural instances of a unit class in the programming language specification.

The Action relies on Schaumont disclosing the above-quoted language of claim 51. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 51. For example, at col. 5, lines 24-30, Schaumont describes "determining the amount of parallelism of operations that process input signals under the form of a vector to output signals." Schaumont later indicates that "[s]aid step of determining the amount of parallelism can preferably comprise determining the amount of parallelism for every data vector and reducing the unspecified communication bandwidth of said data-vector model to a fixed number of communication buses in said data-flow model." (Col. 5, lines 35-40.) Determining parallelism from a vector of signals (as in Schaumont) involves analysis of the signals, not instances of a unit class, and it fails to teach or suggest "transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent execution of units represented by the plural instances of the unit class in the programming language specification," as recited in independent claim 51.

At col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not involve "generating lower-level description for handling concurrent execution of units," and it does not teach or suggest "transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent execution of units represented by the plural instances of the unit class in the programming language specification," as recited in independent claim 51.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members, and "can optionally also be used to perform initialization of other private data." Schaumont goes on to describe "a firing rule and an operative part." This description does not involve "generating lower-level description for handling concurrent execution of units," and it does not teach or suggest "transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent

execution of units represented by the plural instances of the unit class in the programming language specification," as recited in independent claim 51.

At col. 40, lines 17-30, Schaumont describes "mapping of the fixed point library to hardware." This description does not involve "generating lower-level description for handling concurrent execution of units," and it does not teach or suggest "transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent execution of units represented by the plural instances of the unit class in the programming language specification," as recited in independent claim 51.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation, we obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. This description does not involve "generating lower-level description for handling concurrent execution of units," and it does not teach or suggest "transforming the programming language specification into a lower-level specification, wherein the transforming includes generating lower-level description for handling concurrent execution of units represented by the plural instances of the unit class in the programming language specification," as recited in independent claim 51.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 51. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 51.

Dependent Claims 52-57

Claims 52-57 ultimately depend on independent claim 51. Thus, for at least the reasons set forth above with respect to independent claim 51, claims 52-57 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants

respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 52-57.

Independent Claim 58

Independent claim 58 recites:

> a design input module for accepting an algorithmic specification, the algorithmic specification including plural unit calls that map to plural different instances of a unit, thereby indicating parallel execution of the plural unit calls; and
>
> a hardware description language transformer for transforming the algorithmic specification into a lower-level specification, wherein the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls.

Independent claim 58 is directed to a design tool that includes a hardware description language transformer. The transformer is operable to transform the algorithmic specification into a lower-level specification. The transformer is also operable to add code into the lower-level specification for handling the parallel execution of the plural unit calls.

The Action relies on Schaumont disclosing the above-quoted language of claim 58. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 58. For example, at col. 5, lines 24-30, Schaumont describes "determining the amount of parallelism of operations that process input signals under the form of a vector to output signals." Schaumont later indicates that "[s]aid step of determining the amount of parallelism can preferably comprise determining the amount of parallelism for every data vector and reducing the unspecified communication bandwidth of said data-vector model to a fixed number of communication buses in said data-flow model." (Col. 5, lines 35-40.) Determining parallelism from a vector of signals (as in Schaumont) involves analysis of the signals, not different instances of a unit, and it fails to teach or suggest "the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 58.

At col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one

component in the final system implementation." This description does not teach or suggest "the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 58.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members, and "can optionally also be used to perform initialization of other private data." Schaumont goes on to describe "a firing rule and an operative part." This description does not teach or suggest "the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 58.

At col. 40, lines 17-30, Schaumont describes "mapping of the fixed point library to hardware." This description does not teach or suggest "the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 58.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation, we obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. This description does not teach or suggest "the transformer adds code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 58.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 58. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 58.

Dependent Claims 59-62

Claims 59-62 ultimately depend on independent claim 58. Thus, for at least the reasons set forth above with respect to independent claim 58, claims 59-62 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 59-62.

Independent Claim 63

Independent claim 63 recites:

> one or more modules for translating the object-oriented description into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description.

Independent claim 63 is directed to a design tool. One or more modules of the design tool translate an object-oriented description (including an interface) into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description.

The Action relies on Schaumont disclosing the above-quoted language of claim 63. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 63. For example, at col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not involve "translating the object-oriented description into a lower-level specification" or "plural ports specified according to defined semantics," and it does not teach or suggest "one or more modules for translating the object-oriented description into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description," as recited in independent claim 63.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members (which are "pointers to communication queues" (col. 15, lines 21-22)), and "can optionally also be used to

perform initialization of other private data." The parameters of the constructor include "a list of the queues that are used for communication." (Col. 15, line 29.) Schaumont goes on to describe "a firing rule and an operative part." Merely specifying communication queues for a block with private members (as in Schaumont) does not involve "plural ports specified according to defined semantics," and it does not teach or suggest "one or more modules for translating the object-oriented description into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description," as recited in independent claim 63. Similarly, including a list of communication queues as parameters for a constructor (as in Schaumont) also fails to teach or suggest the above-quoted language of claim 63.

At col. 40, lines 17-30, Schaumont describes "mapping of the fixed point library to hardware." This description does not involve "translating the object-oriented description into a lower-level specification" or "plural ports specified according to defined semantics," and it does not teach or suggest "one or more modules for translating the object-oriented description into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description," as recited in independent claim 63.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation" to "obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. In this section, Schaumont references files 3.3 and 3.4, in which private members are declared for flow buffers, and a constructor includes parameters "_symb1" and "_symb2" for the flow buffers. Again, merely including a list of communication queues as parameters for a constructor (as in Schaumont) does not involve "translating the object-oriented description into a lower-level specification" or "plural ports specified according to defined semantics," and it does not teach or suggest "one or more modules for translating the object-oriented description into a lower-level specification having plural ports specified according to defined semantics for the interface of the object-oriented description," as recited in independent claim 63.

Elsewhere, Schaumont describes using signal data types and a signal flowgraph "that collects all behavior that must be executed during one clock cycle," where the behavior contains "[a] set of inputs and outputs that relate signals to output and input queues." (Col. 25, lines 54-61.) "[A] signal flowgraph object connects local behavior (the signals) to the system through communication queues. In hardware, the indication of input and output signals also results in ports on your resulting circuit." (Col. 25, lines 62-65; *see also* col. 38, lines 38-54.) Even if, for the sake of argument, Schaumont describes determining ports of a circuit, such determination uses signal data types and a detailed signal flowgraph (*id.*), which leads away from the above-quoted language of claim 63.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 63. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 63.

Dependent Claims 64-68

Claims 64-68 ultimately depend on independent claim 63. Thus, for at least the reasons set forth above with respect to independent claim 63, claims 64-68 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 64-68.

Independent Claim 69

Independent claim 69 recites:

> receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, the lower-level specification having plural ports specified according to defined semantics for the interface of the programming language specification.

Independent claim 69 is directed to transforming a programming language specification (including an interface) into a lower-level specification. The lower-level specification has plural ports specified according to defined semantics for the interface of the programming language specification.

The Action relies on Schaumont disclosing the above-quoted language of claim 69. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 69. For example, at col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not involve "a lower-level specification produced by transforming the programming language specification into the lower-level specification" or "plural ports specified according to defined semantics," and it does not teach or suggest "receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, the lower-level specification having plural ports specified according to defined semantics for the interface of the programming language specification," as recited in independent claim 69.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members(which are "pointers to communication queues" (col. 15, lines 21-22)), and "can optionally also be used to perform initialization of other private data." The parameters of the constructor include "a list of the queues that are used for communication." (Col. 15, line 29.) Schaumont goes on to describe "a firing rule and an operative part." Merely specifying communication queues for a block with private members (as in Schaumont) does not involve "plural ports specified according to defined semantics," and it does not teach or suggest "receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, the lower-level specification having plural ports specified according to defined semantics for the interface of the programming language specification," as recited in independent claim 69. Similarly, including a list of communication queues as parameters for a constructor (as in Schaumont) also fails to teach or suggest the above-quoted language of claim 69.

At col. 40, lines 17-30, Schaumont describes "mapping of the fixed point library to hardware." This description does not involve "a lower-level specification produced by transforming the programming language specification into the lower-level specification" or "plural ports specified according to defined semantics," and does not teach or suggest "receiving a lower-level specification produced by transforming the programming language specification

into the lower-level specification, the lower-level specification having plural ports specified according to defined semantics for the interface of the programming language specification," as recited in independent claim 69.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation" to "obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. In this section, Schaumont references files 3.3 and 3.4, in which private members are declared for flow buffers, and a constructor includes parameters "_symb1" and "_symb2" for the flow buffers. Again, merely including a list of communication queues as parameters for a constructor (as in Schaumont) does not involve "a lower-level specification produced by transforming the programming language specification into the lower-level specification" or "plural ports specified according to defined semantics," and it does not teach or suggest "receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, the lower-level specification having plural ports specified according to defined semantics for the interface of the programming language specification," as recited in independent claim 69.

Elsewhere, Schaumont describes using signal data types and a signal flowgraph "that collects all behavior that must be executed during one clock cycle," where the behavior contains "[a] set of inputs and outputs that relate signals to output and input queues." (Col. 25, lines 54-61.) "[A] signal flowgraph object connects local behavior (the signals) to the system through communication queues. In hardware, the indication of input and output signals also results in ports on your resulting circuit." (Col. 25, lines 62-65; *see also* col. 38, lines 38-54.) Even if, for the sake of argument, Schaumont describes determining ports of a circuit, such determination uses signal data types and a detailed signal flowgraph (*id.*), which leads away from the above-quoted language of claim 69.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 69. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 69.

Dependent Claims 70-71

Claims 70-71 ultimately depend on independent claim 69. Thus, for at least the reasons set forth above with respect to independent claim 69, claims 70-71 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 70-71.

Independent Claim 72

Independent claim 72 recites:

> providing a programming language specification, the programming language specification including plural unit calls that map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls; and
>
> receiving a lower-level specification produced by transforming the programming language specification into the lower-level specification, wherein the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls.

Independent claim 72 is directed to transforming a programming language specification into a lower-level specification. Code is added into the lower-level specification for handling the parallel execution of plural unit calls.

The Action relies on Schaumont disclosing the above-quoted language of claim 72. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 72. For example, at col. 5, lines 24-30, Schaumont describes "determining the amount of parallelism of operations that process input signals under the form of a vector to output signals." Schaumont later indicates that "[s]aid step of determining the amount of parallelism can preferably comprise determining the amount of parallelism for every data vector and reducing the unspecified communication bandwidth of said data-vector model to a fixed number of communication buses in said data-flow model." (Col. 5,

lines 35-40.) Determining parallelism from a vector of signals (as in Schaumont) involves analysis of the signals, not different instances of a unit class, and it fails to teach or suggest "the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 72.

At col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not involve "parallel execution of the plural unit calls," and it does not teach or suggest "the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 72.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members, and "can optionally also be used to perform initialization of other private data." Schaumont goes on to describe "a firing rule and an operative part." This description does not involve "parallel execution of the plural unit calls," and it does not teach or suggest "the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 72.

At col. 40, lines 17-30, Schaumont describes "mapping of the fixed point library to hardware." This description does not involve "parallel execution of the plural unit calls," and it does not teach or suggest "the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 72.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation, we obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a

subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. This description does not involve "parallel execution of the plural unit calls," and it does not teach or suggest "the transforming includes adding code into the lower-level specification for handling the parallel execution of the plural unit calls," where the plural unit calls "map to plural different instances of a unit class, thereby indicating parallel execution of the plural unit calls," as recited in independent claim 72.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 72. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 72.

Dependent Claims 73-74

Claims 73-74 ultimately depend on independent claim 72. Thus, for at least the reasons set forth above with respect to independent claim 72, claims 73-74 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 73-74.

Independent Claim 75

Independent claim 75 recites:

> receiving an object-oriented description, the object-oriented description including native programming language code, wherein the object-oriented description specifies structural details of a design unit for synthesis with a design tool into a HDL description of the design unit; and
>
> compiling the object-oriented description with a native programming language compiler for algorithmic validation of the design unit independent of the synthesis.

Independent claim 75 is directed to compiling an object-oriented description with a native programming language compiler for algorithmic validation of a design unit independent of synthesis.

The Action relies on Schaumont disclosing the above-quoted language of claim 75. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 75. For example, at col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not involve "receiving an object-oriented description, the object-oriented description including native programming language code" or "compiling the object-oriented description with a native programming language compiler," and it does not teach or suggest "receiving an object-oriented description, the object-oriented description including native programming language code" and "compiling the object-oriented description with a native programming language compiler for algorithmic validation of the design unit independent of the synthesis," as recited in independent claim 75.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members, and "can optionally also be used to perform initialization of other private data." Schaumont goes on to describe "a firing rule and an operative part." This description does not teach or suggest the above-quoted language of claim 75.

At col. 63, lines 1-13, Schaumont describes "the target architecture, which must be chosen such that the requirements are met," and "a programming approach to implementation," "in which the system is modeled in C++." At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation, we obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. Elsewhere, Schaumont clarifies that the C++ it describes uses the "OCAPI" library, which is "a C++ library for the design of digital systems." (Col. 8, lines 7-8; *see also* col. 7, lines 21-29; col. 9, line 30 to col. 10, line 35.) Using a version of C++ with

specialized library for the design of digital systems (as in Schaumont) leads away from the above-cited "native programming language code" language of claim 75.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 75. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 75.

Dependent Claims 76-78

Claims 76-78 ultimately depend on independent claim 75. Thus, for at least the reasons set forth above with respect to independent claim 75, claims 76-78 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 76-78.

Independent Claim 79

Independent claim 79 recites:

> automatically creating one or more local signals of a first design unit, the first design unit including an instance of a second design unit without explicitly specifying connection logic between the one or more local signals and corresponding ports of the second design unit; and
>
> automatically mapping the one or more local signals to the corresponding ports of the second design unit.

Independent claim 79 is directed to implementing a hierarchical relationship between a first design unit and a second design unit. One or more local signals of the first design unit are automatically mapped to corresponding ports of the second design unit.

The Action relies on Schaumont disclosing the above-quoted language of claim 79. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 79. For example, at col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not involve "mapping the one or more local signals" or "corresponding ports," and it does not teach or suggest "automatically mapping the one or more

local signals to the corresponding ports of the second design unit," as recited in independent claim 79.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members, and "can optionally also be used to perform initialization of other private data." Schaumont goes on to describe "a firing rule and an operative part." This description does not involve "corresponding ports," and it does not teach or suggest "automatically mapping the one or more local signals to the corresponding ports of the second design unit," as recited in independent claim 79.

At col. 40, lines 17-30, Schaumont describes "mapping of the fixed point library to hardware." This description does not involve "corresponding ports," and it does not teach or suggest "automatically mapping the one or more local signals to the corresponding ports of the second design unit," as recited in independent claim 79.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation" to "obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. This description does not involve "corresponding ports," and it does not teach or suggest "automatically mapping the one or more local signals to the corresponding ports of the second design unit," as recited in independent claim 79.

Elsewhere, Schaumont describes using signal data types and a signal flowgraph "that collects all behavior that must be executed during one clock cycle," where the behavior contains "[a] set of inputs and outputs that relate signals to output and input queues." (Col. 25, lines 54-61.) "[A] signal flowgraph object connects local behavior (the signals) to the system through communication queues. In hardware, the indication of input and output signals also results in ports on your resulting circuit." (Col. 25, lines 62-65; *see also* col. 38, lines 38-54.) Even if, for the sake of argument, Schaumont describes determining ports of a circuit, such determination

uses signal data types and a detailed signal flowgraph (*id.*), which leads away from the above-quoted language of claim 79.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 79. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 79.

Dependent Claims 80-83

Claims 80-83 ultimately depend on independent claim 79. Thus, for at least the reasons set forth above with respect to independent claim 79, claims 80-83 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 80-83.

Independent Claim 84

Independent claim 84 recites:

> accepting a programming language specification of a system including software and hardware, wherein the programming language specification follows the same syntax for the software and the hardware; and
>
> transforming at least part of the programming language specification into a lower-level specification.

Independent claim 84 is directed to creating a system including software and hardware. At least part of a programming language specification is transformed into a lower-level specification.

The Action relies on Schaumont disclosing the above-quoted language of claim 84. Applicants respectfully disagree.

The Action cites various passages in Schaumont, which are each addressed below, against the above-quoted language of claim 84. For example, at col. 8, lines 59-61, Schaumont states that "[t]he system machine model that is used is a set of concurrent processes." Schaumont goes on to state that "[e]ach process translates to one component in the final system implementation." This description does not teach or suggest the above-cited language of claim 84.

At col. 15, line 54, to col. 16, line 17, Schaumont describes a constructor that passes the name of an object to the "base" class it inherits from, initializes private members, and "can optionally also be used to perform initialization of other private data." Schaumont goes on to describe "a firing rule and an operative part." This description does not teach or suggest the above-cited language of claim 84.

At col. 69, line 31, to col. 71, line 24, Schaumont describes an overall design flow in which an "initial specification is an architecture model, constructed in C++," constructing "an architectural model out of it," and, "relying on code generation, we obtain a synthesizable architecture model." Schaumont goes on to "consider the construction of an actor," which "is a subalgorithm out of a dataflow system model," and describe a "dataflow system" that "is constructed out of such actors." Schaumont goes on to describe an "architecture model" that "expresses the behavior of the algorithmic model in terms of operations onto hardware." Schaumont also describes how "the output of the code generation process" is indicated. Elsewhere, Schaumont clarifies that the C++ it describes uses the "OCAPI" library, which is "a C++ library for the design of digital systems." (Col. 8, lines 7-8; *see also* col. 7, lines 21-29; col. 9, line 30 to col. 10, line 35.) Using a version of C++ with specialized library for the design of digital systems (as in Schaumont) involves programming for a hardware system, and it leads away from the above-cited language of claim 84.

Therefore, Applicants respectfully submit that Schaumont does not show each and every element as set forth in claim 84. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from independent claim 84.

Dependent Claims 85-87

Claims 85-87 ultimately depend on independent claim 84. Thus, for at least the reasons set forth above with respect to independent claim 84, claims 85-87 should also be in condition for allowance. These claims are also independently patentable. Accordingly, Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn from dependent claims 85-87.

Request for Interview

Applicants believe the application to be allowable. If any issues remain, however, the Examiner is formally requested to contact the undersigned attorney at (503) 226-7391 prior to issuance of the next communication in order to arrange a telephonic interview.

This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.

Conclusion

The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By _____
Justin D. Wagner
Registration No. 54,519

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446

cc: Docketing